Sistema de Gestión de Cursos

Introducción

Este documento detalla la arquitectura de la aplicación creada para la resolución del trabajo práctico. La misma requiere crear una aplicación para la gestión de trámites y consultas de la materia.

Como la idea es proveer un panorama amplio de la arquitectura, los diagramas y explicaciones no tienen un gran nivel de detalle, sino que muestran información general sobre la arquitectura y el diseño.

Índice

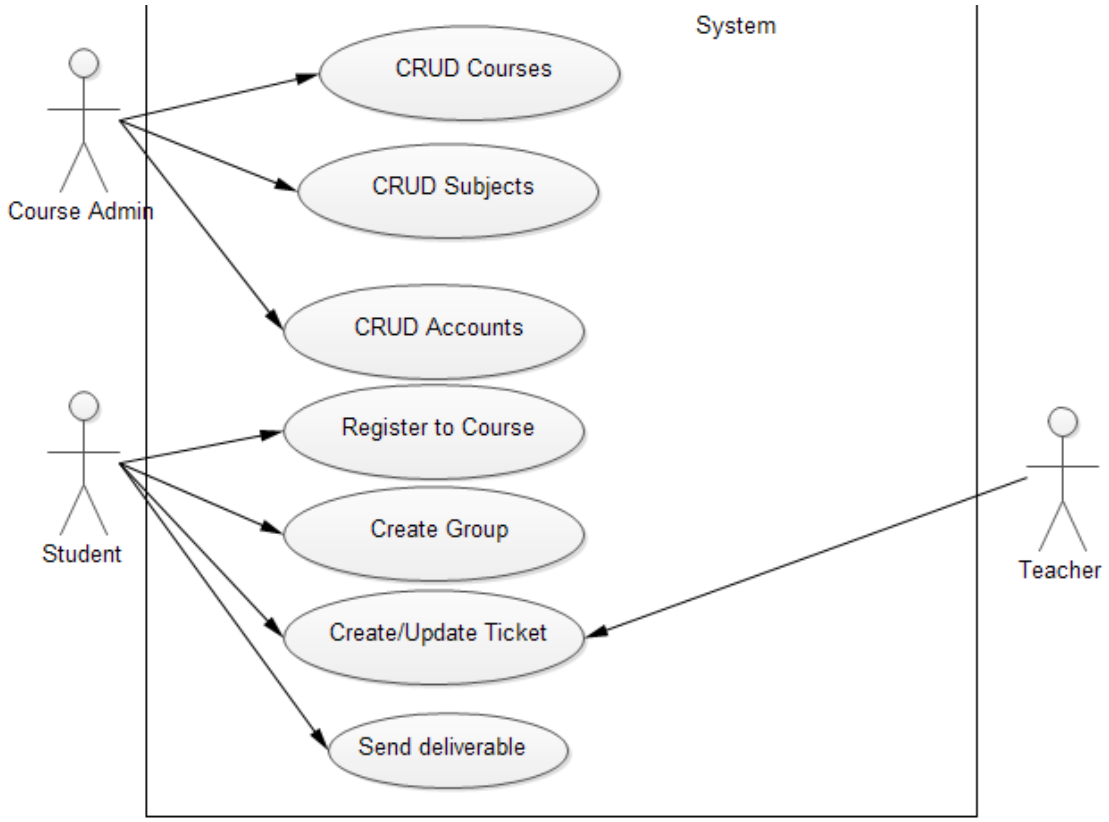
Vista de Casos de Uso

En esta sección se mencionan los casos de uso principales de la aplicación.

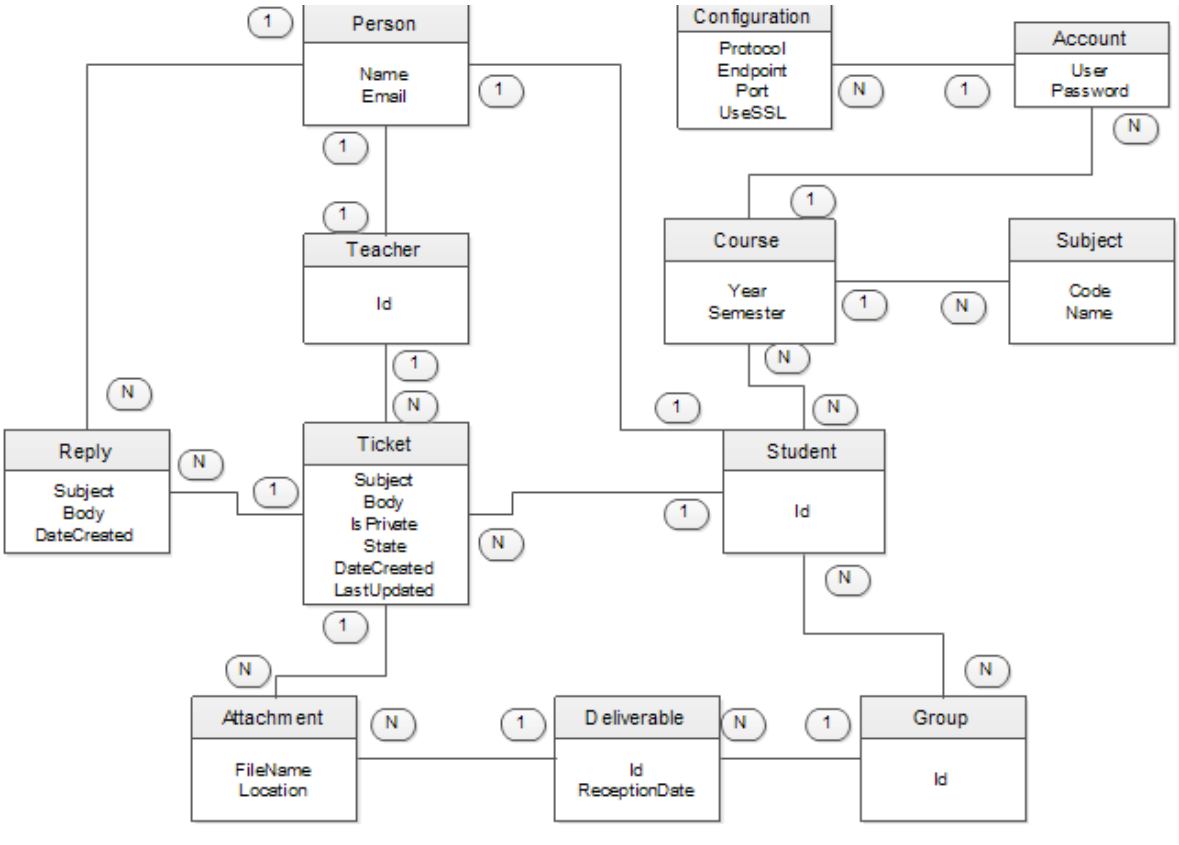
1. Configurar cuentas de e-mail de un curso de una materia.
2. Administrar materias y cursos (crear, modificar y eliminar).
3. Procesamiento de e-mails bajo determinadas reglas, las cuales permiten:
    - Dar de alta de alumnos en un curso.
    - Registrar entregables de alumnos, incluyendo la persistencia en disco de archivos adjuntos a e-mails.
    - Crear nuevos tickets.
    - Responder a tickets existentes.
    - Crear grupos en un curso y agregar alumnos al grupo.
4. Monitoreo de estado de los temas de discusión.
5. Ver estadísticas (alumnos inscriptos, grupos asignados, TPs recibidos, e-mails recibidos, estados de temas de discusión, número de e-mails que no cumplen con las reglas).

Nota: Los ítems 4 y 5 no son parte del scope del proyecto (al menos en esta versión). Es decir, la información necesaria para cumplir con dichos casos de uso será guardada, pero el sistema no permitirá consultarla.

El siguiente diagrama de casos de uso contempla aquellos que están dentro del alcance del proyecto.



Adicionalmente, incluimos a continuación una versión simplificada del diagrama de entidad-interrelación, ya que consideramos que puede ser útil para facilitar la comprensión de los casos de uso solicitados por la aplicación.



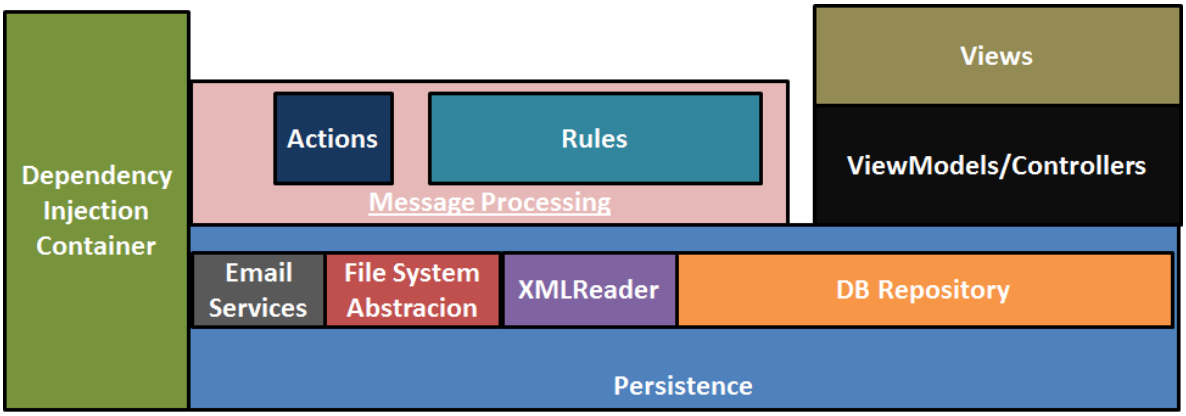
Vista de Desarrollo

Para el desarrollo de la aplicación se utilizara:

- C# como lenguaje de programación.
- Visual Studio 2010 como ambiente de desarrollo.
- Entity Framework 5.0 como ORM.
- Moq y Moles para crear mocks.
- Unity como contenedor de inyección de dependencias.
- Una biblioteca a definir para la obtención de e-mail a través de POP3.

Vista Lógica

La siguiente figura muestra una vista en capas de la aplicación.



Desde el comienzo del proceso de desarrollo se buscará trabajar con abstracciones, no implementaciones concretas, a fin de aumentar el desacople entre componentes, lo que provee mayor facilidad para cambiar los mismos y realizar tests unitarios.

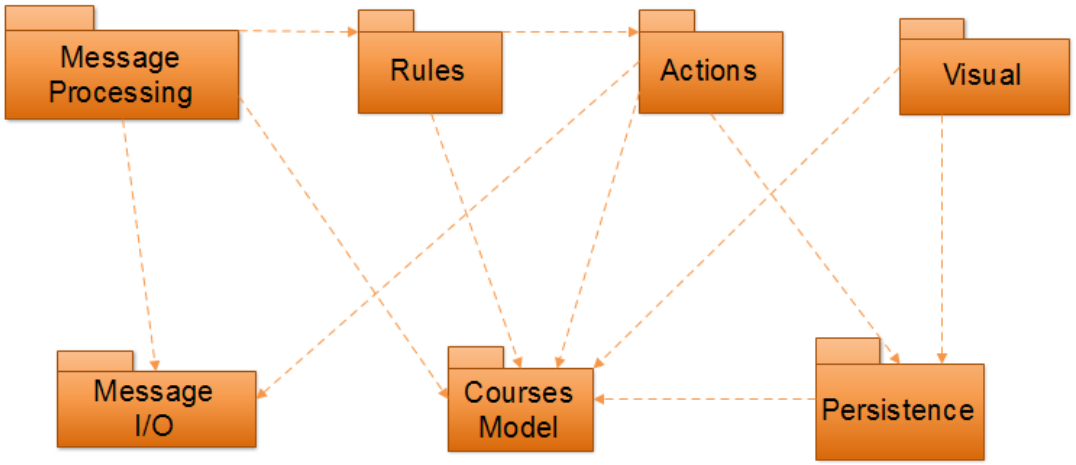
Un ejemplo del desacoplamiento buscado se ve en la persistencia. La persistencia a los diferentes medios se realizará mediante abstracciones. En la aplicación a desarrollar, los repositorios ejecutan consultas contra una base de datos. Implementando correctamente la interfaz de los mismos, se podría conseguir actualizar la aplicación para utilizar Google Spreadsheets. De igual forma, se utiliza un patrón de "presentación separada", no para utilizar el mismo controller/viewmodel para distintas vistas, sino para mantener la lógica desacoplada de las vistas.

Para lograr el desacoplamiento de componentes, cada clase recibe en su constructor interfaces, no implementaciones concretas. Para facilitar la construcción de estos objetos y poder modificar de manera simple y rápida la implementación a utilizar, se decidió utilizar un DependencyInjectionContainer, UnityContainer. El mismo se configura mediante un Bootstrapper.

Diagrama de paquetes

Los siguientes son los diagramas de paquetes resultantes pensados para la aplicación (sin incluir los paquetes de pruebas).

Este es el diagrama que muestra los paquetes más relevantes y sus relaciones de utilización:



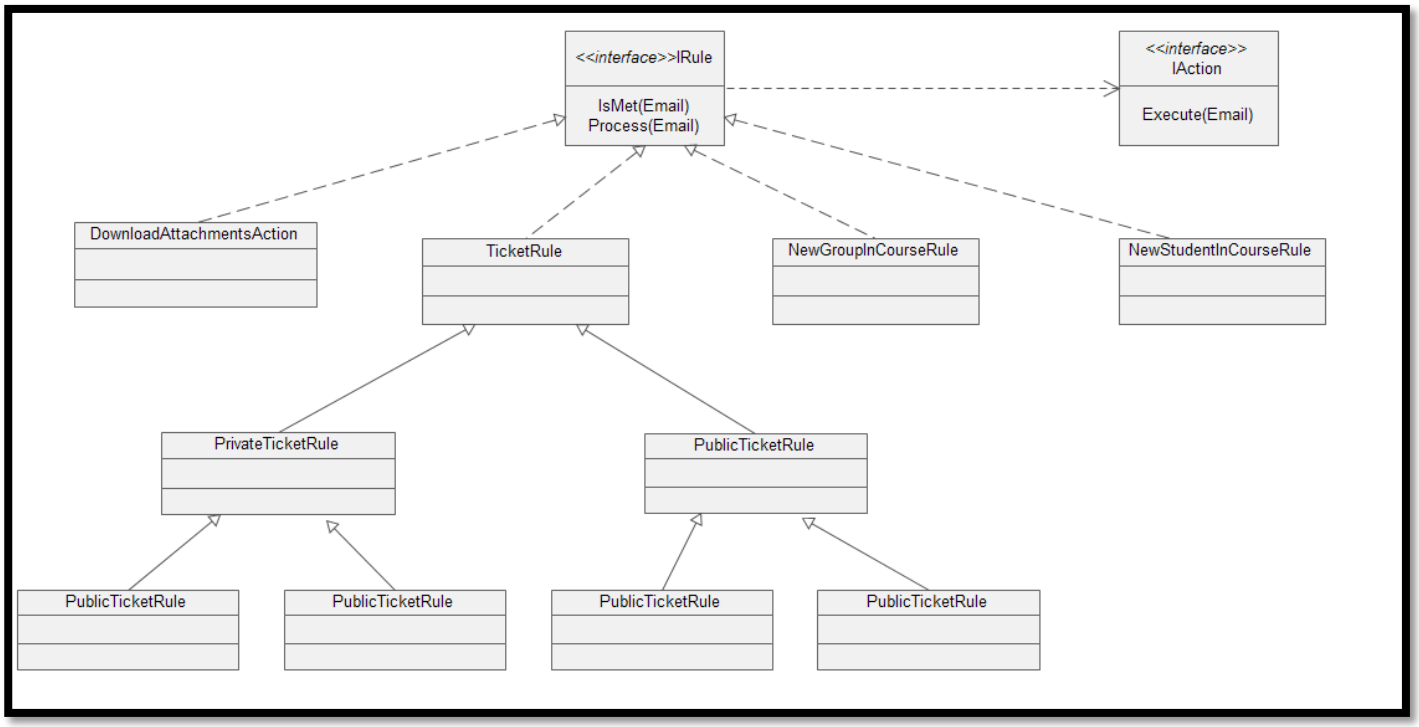
El contenido de cada paquete se explica a continuación:

- **CoursesModel:**Contiene a las clases que representan al dominio de la aplicación.
- **Persistence:**Contiene las clases necesarias para guardar los datos generados por el procesamiento de e-mails y recuperarlos.
- **Visual:**Contiene las interfaces de las vistas y la implementación de las mismas así como también implementación de los controladores para cada vista y las interfaces de los mismos.
- **MessageProcessing:** Contiene a los componentes necesarios para procesar los mensajes recuperados a través de las diferentes reglas.
- **Message I/O:** Contiene a los componentes necesarios para enviar y recibir mensajes.

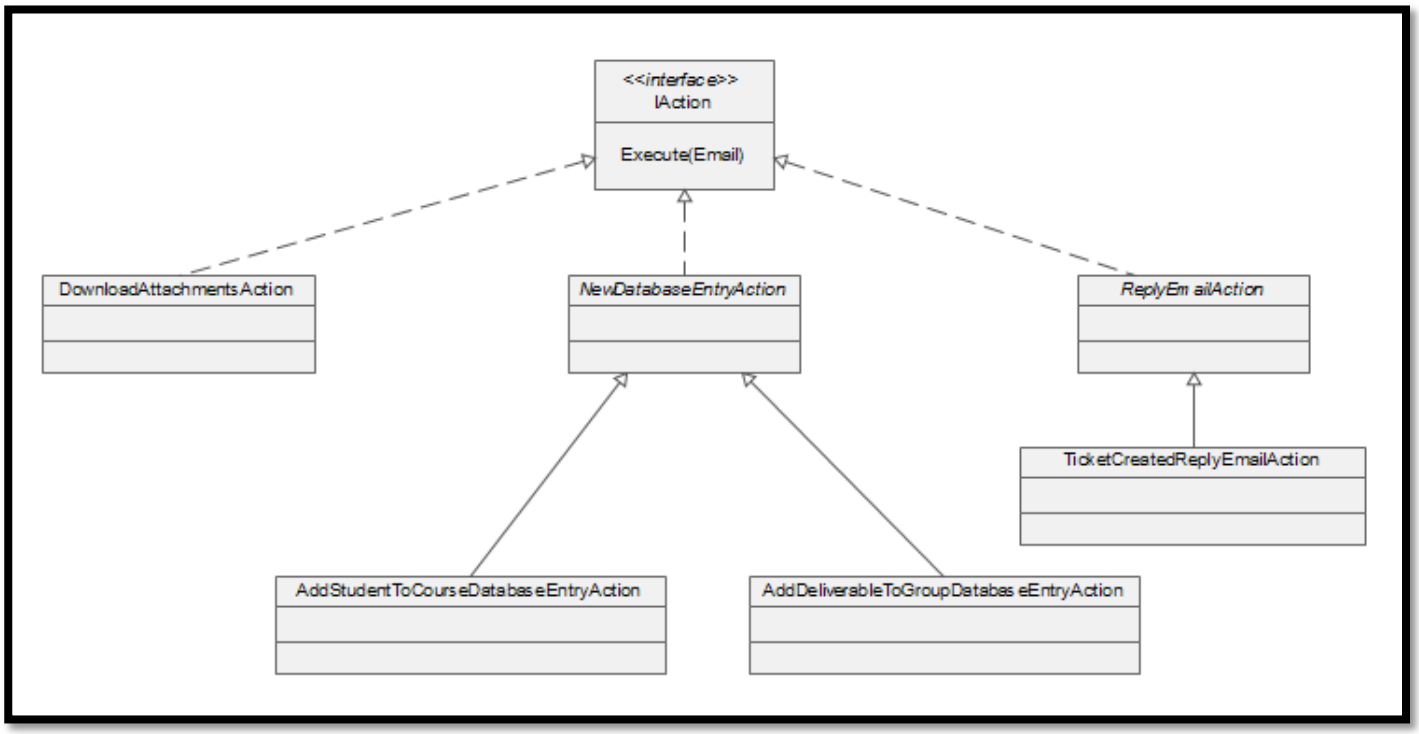
Diagramas de Clases

A continuación se muestran los diagramas de clases diseñados para la aplicación.

Se muestran dos diagramas parciales, que contienen las jerarquías de Reglas y Acciones; y por último se muestra el diagrama de clases general.



(Diagrama de Clases Parcial – Jerarquía de Reglas)

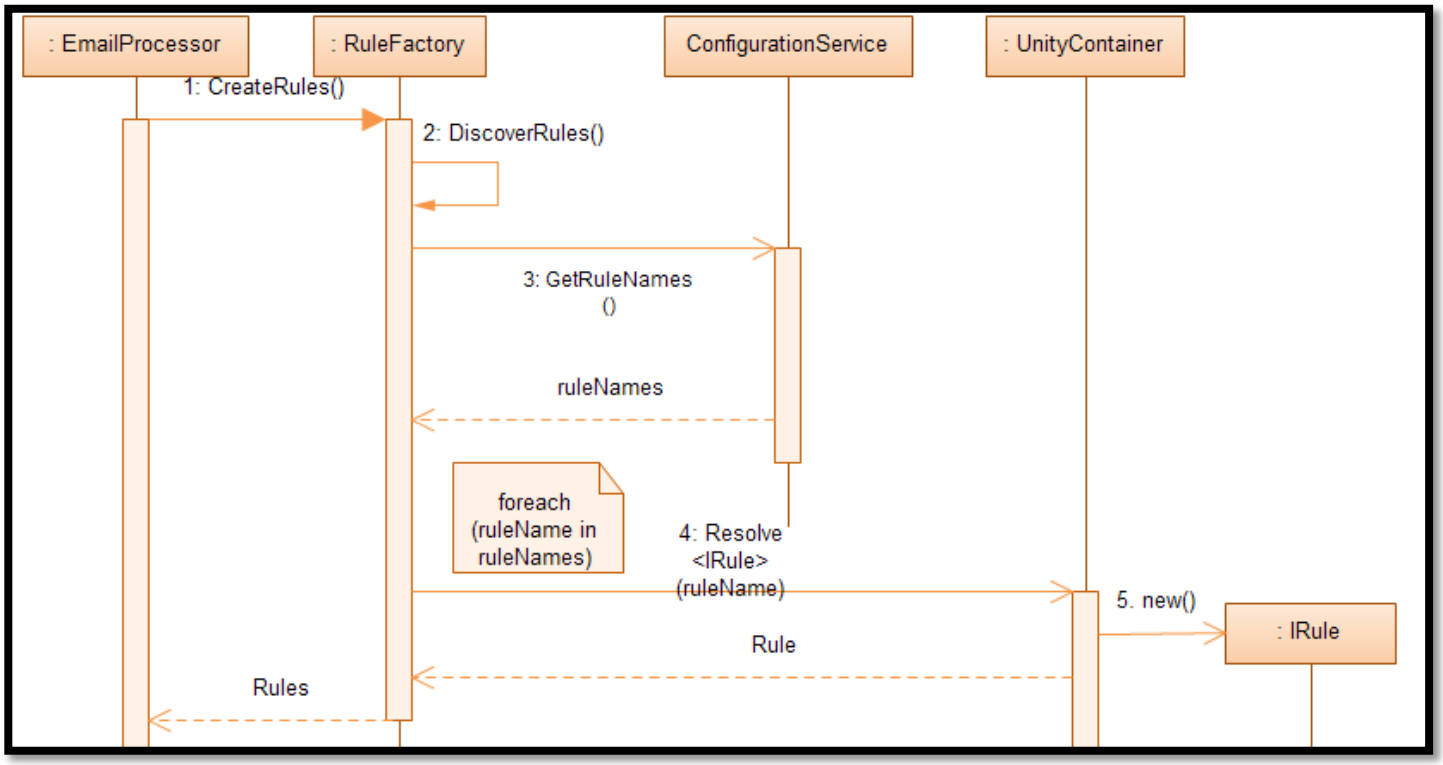


(Diagrama de Clases Parcial – Jerarquía de Acciones)

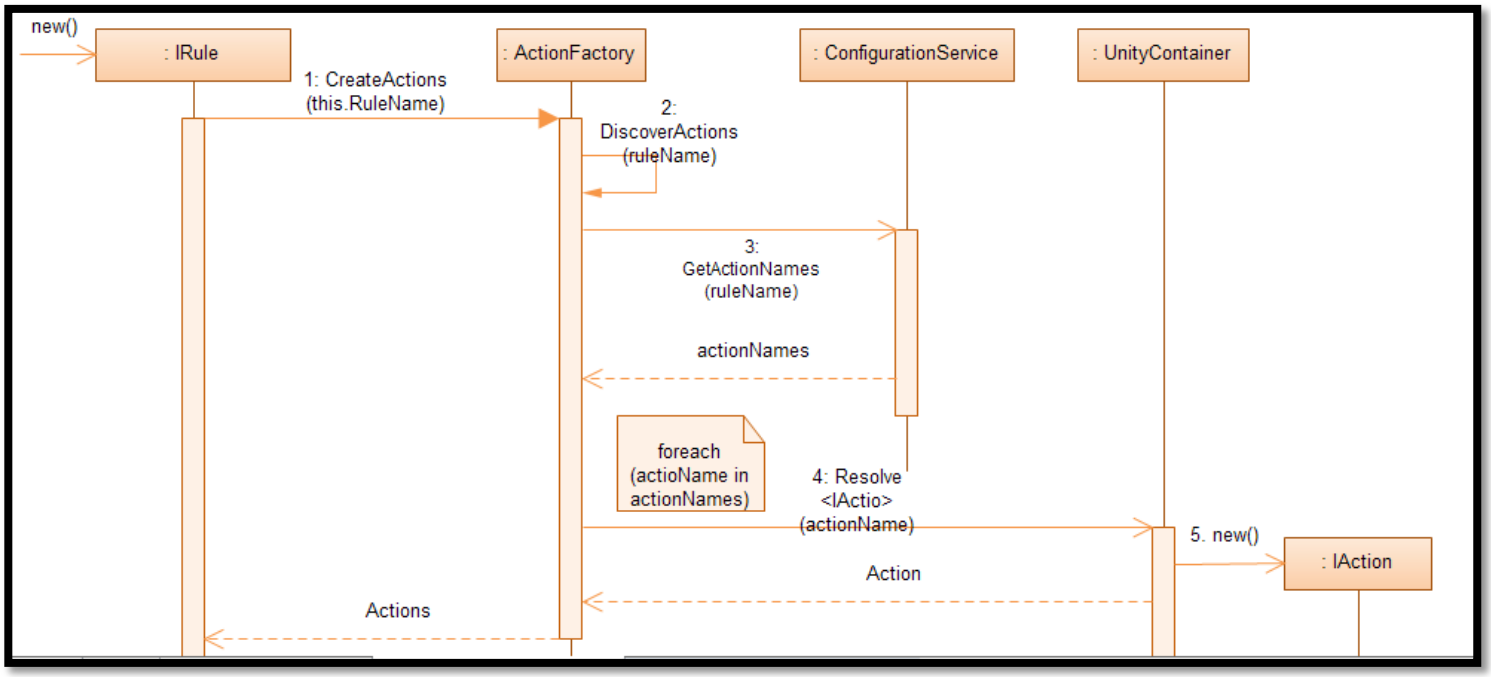
(Diagrama de Clases General)

Diagramas de Secuencia

Se detallan los diagramas más relevantes de la aplicación. Los mismos corresponden a las secuencias de obtención de reglas y acciones a partir del procesado de emails.



(RuleBootstrapSequence)



(ActionBootstrapSequence)

Vista de procesos

La aplicación cuenta con:

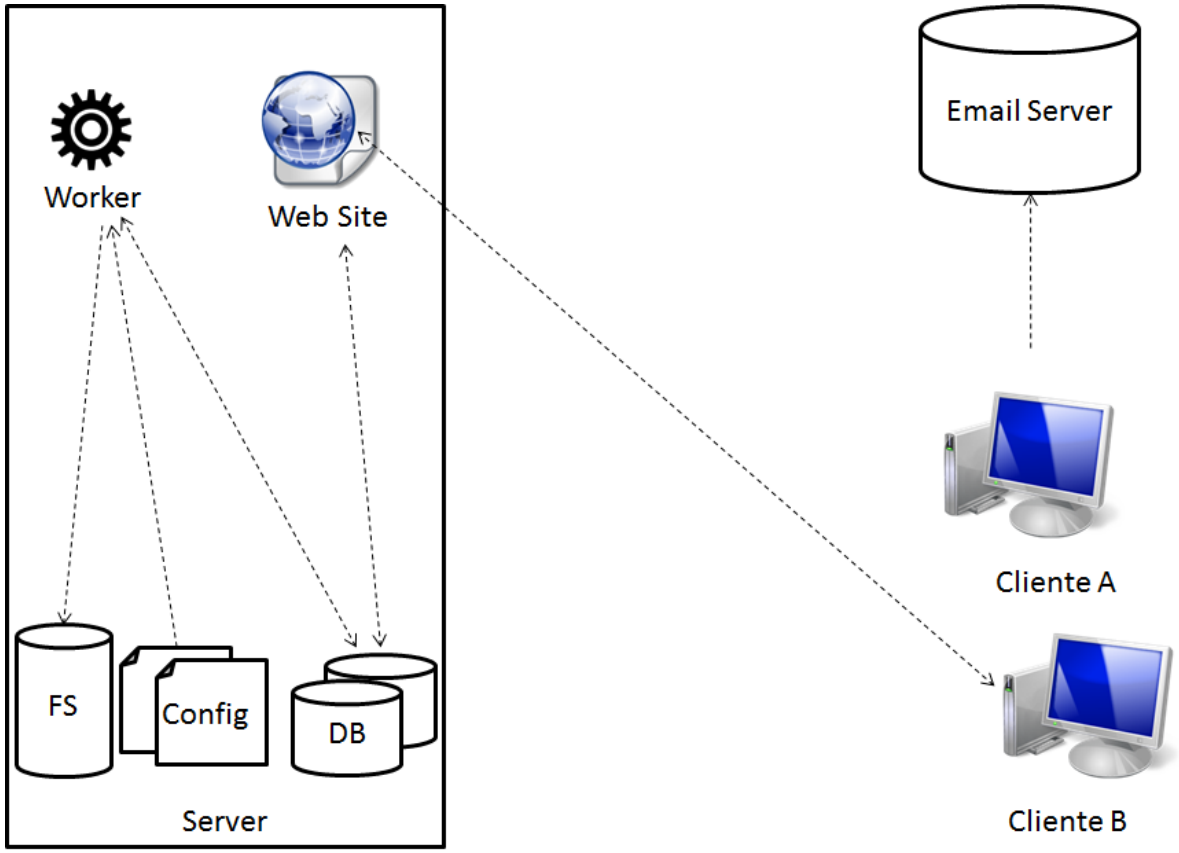
- Un proceso en un loop infinito (worker)que continuamente recuperara los e-mails nuevos de las cuentas configuradas para una materia y los procesará.
- Un proceso para una aplicación de escritorio / web, que permitirá adminstrar materias, cursos, cuentas de e-mail y configuración de cuentas.

Estos dos procesos no necesitan mecanismos de sincronización ya que en ningún momento interactúan directamente entre sí, sino que escriben la base de datos.

Vista física

A continuación se muestra la distribución propuesta para la aplicación suponiendo una interfaz web para la administración de cursos. Otra posibilidad (no ilustrada aquí por ser más simple), involucra a una aplicación de escritorio que suplante a la interfaz web.

Nota: Esta distribución considera la demo de la aplicación (por cuestiones de tiempo y más importante aún, presupuestarias). Si esta aplicación fuera puesta en producción se recomendaría utilizar servidores separados para el Worker, la aplicación Web y la base de datos, asi como también utilizar un servicio de Storage as a Service.


Worker
Web Site
Email Server
Cliente A
FS
Config
DB
Server
Cliente B